Final Exam Review -2022

AW

Logistics

- The exam is **Tuesday, May 03, 2022**
- Official time is 2:00 pm - 4:30 pm
- Open book, open notes
- Can use computer and web
- When requested must provide intermediate steps for full credit
- Good luck!

Questions and Grading

Composition:

- Undergrad/grad section – very similar to HWs
  - 4 open problem questions
  - T/F-MC section
- Grad only section – not in HWs but in lectures
  - 1 open problem

Grading:

- Grades given in [ ] for undergrad, multiplier fraction for grad students is { 4/5}
- Distribution: 25,25,25,25 MC/TF 5 each – total 120, max earned 100
- Grad only question is 24 pts
- Questions have partial credit

Lecture Overview

1. MT composition
2. Computational Graph
3. Apply convolutional filter
4. Autoencoder
5. Network Training
6. True/False and MC questions
7. What happens if …

Questions and Grading

Composition:

- Undergrad/grad section – very similar to HWs
  - 4 open problem questions
  - T/F-MC section
- Grad only section – not in HWs but in lectures
  - 1 open problem

Grading:

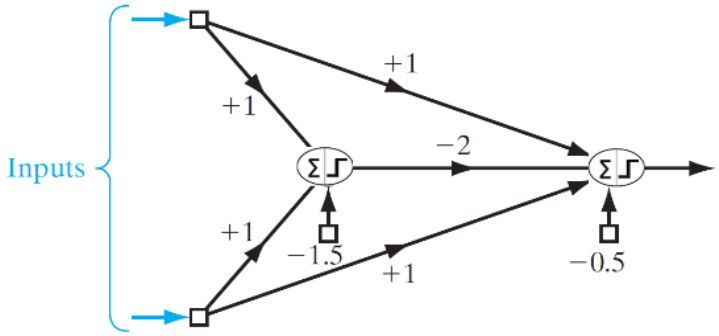
- Grades given in [ ] for undergrad, multiplier fraction for grad students is { 4/5}
- Distribution: 25,25,25,25 MC/TF 5 each – total 120, max earned 100
- Grad only question is 24 pts
- Questions have partial credit

Lecture Overview

1. MT composition
2. Computational Graph
3. Apply convolutional filter
4. Autoencoder
5. Network Training
6. True/False and MC questions
7. What happens if …

Computational Graph For the Network

Construct a computational graph for the following network:

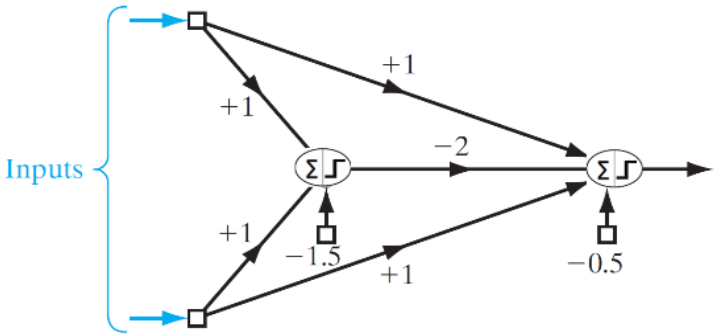


Solution:

- The input data for the graph is
  - Input vector $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$,
  - The weight vector of hidden layer $\vec{w}_h = \begin{pmatrix} w_1 \\ w_2 \\ b_1 \end{pmatrix}$,
  - Weight vector of output layer $\vec{w}_o = \begin{pmatrix} w_3 \\ w_4 \\ w_5 \\ b_2 \end{pmatrix}$

Reshaping for Computational Graph

Construct a computational graph for the following network:



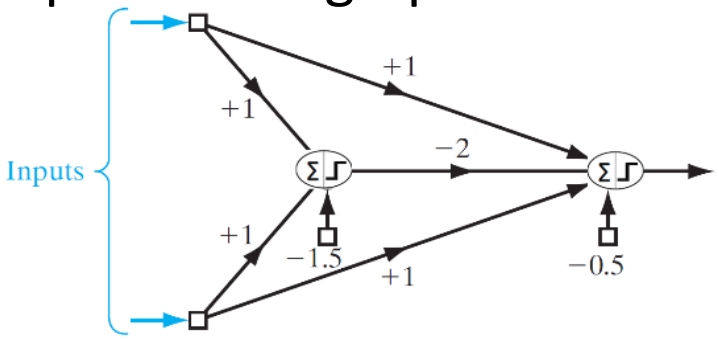
Solution.

We need to do 3 reshape operations to add bias:

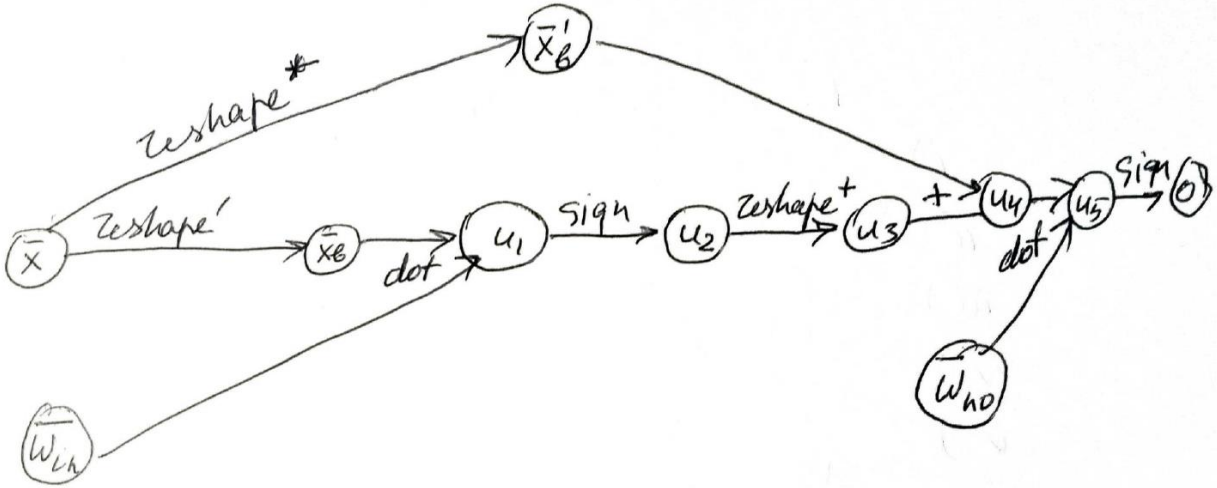
- one reshape operation reshapes input $\vec{x}$ to $\vec{x}_b$ embedding 2-dim vector $\vec{x}$ into 3 dim vector $\vec{x}_b$ by setting $x_3 = 1$ in order to accommodate hidden neuron;
- Another reshape operation embeds 2-dim input vector $\vec{x}$ into 4 dim vector $\vec{x}'_b$ by setting $x'_1 = 0, x'_{i+1} = x_i$ for all $3 > i > 0$ and setting $x_4 = 1$ in order to accommodate output neuron
- Finally last reshape operation takes output of hidden neuron (real value u_2) and converts it into 4-dim vector $\vec{u}_3$ with $u_{31} = u_2$ and all other coordinates 0.

Computational Graph For the Network

Construct a computational graph for the following network:



Solution. Computational graph:



Lecture Overview

1. MT composition
2. Computational Graph
3. **Apply convolutional filter**
4. Autoencoder
5. Network Training
6. True/False and MC questions
7. What happens if …

Convolutional Filtering Input

Compute the convolution of the input volume matrix with the vertical edge detection filter. Use a stride of 3 with padding (2).

$$\text{Input matrix } \mathrm{A} = \begin{pmatrix} 3 & 4 & 1 & 1 & 0 & 4 & 5 & 1 \\ 0 & 4 & 2 & 2 & 0 & 5 & 4 & 0 \\ 5 & 4 & 0 & 0 & 1 & 5 & 4 & 2 \\ 4 & 4 & 0 & 2 & 1 & 3 & 5 & 0 \\ 4 & 4 & 3 & 3 & 1 & 5 & 5 & 3 \\ 3 & 5 & 3 & 0 & 2 & 3 & 5 & 2 \\ 5 & 5 & 4 & 3 & 3 & 5 & 5 & 4 \\ 0 & 5 & 5 & 0 & 0 & 5 & 5 & 0 \end{pmatrix}$$

$$\text{Vertical edge detection filter } \mathrm{F} = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$

Convolutional Filter and Blocks

Solution: Padded matrix below has dimensions 12×12

$$A_{pad} = \left(\begin{array}{ccc|ccc|ccc|ccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 4 & 1 & 1 & 0 & 4 & 5 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 4 & 2 & 2 & 0 & 5 & 4 & 0 & 0 & 0 \\ 0 & 0 & 5 & 4 & 0 & 0 & 1 & 5 & 4 & 2 & 0 & 0 \\ 0 & 0 & 4 & 4 & 0 & 2 & 1 & 3 & 5 & 0 & 0 & 0 \\ \hline 0 & 0 & 4 & 4 & 3 & 3 & 1 & 5 & 5 & 3 & 0 & 0 \\ 0 & 0 & 3 & 5 & 3 & 0 & 2 & 3 & 5 & 2 & 0 & 0 \\ 0 & 0 & 5 & 5 & 4 & 3 & 3 & 5 & 5 & 4 & 0 & 0 \\ \hline 0 & 0 & 0 & 5 & 5 & 0 & 0 & 5 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

Stride 3 matches the size of filter matrix so when we shift 3×3 matrix by 3×3 blocks. We must apply filter to every submatrix of the matrix to get the result

Convolutional Filter Application

Solution: for example when we apply the filter

$$\mathrm{F} = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$ to block $A_{23} = \begin{pmatrix} 0 & 5 & 4 \\ 1 & 5 & 4 \\ 1 & 3 & 5 \end{pmatrix}$ we obtain the value $\sum \begin{pmatrix} 1 \times 0 & 0 \times 5 & -1 \times 4 \\ 1 \times 1 & 0 \times 5 & -1 \times 4 \\ 1 \times 1 & 0 \times 3 & -1 \times 5 \end{pmatrix}$

$$= 0 + 0 - 4 + 1 + 0 - 4 + 1 + 0 - 5 = -11$$

By applying to all blocks we get the result

$$\begin{pmatrix} -3 & 3 & -5 & 1 \\ -9 & 8 & -11 & 2 \\ -12 & 8 & -9 & 9 \\ 0 & 5 & -5 & 0 \end{pmatrix}$$

Lecture Overview

1. MT composition
2. Computational Graph
3. Apply convolutional filter
4. Autoencoder
5. Network Training
6. True/False and MC questions
7. What happens if …

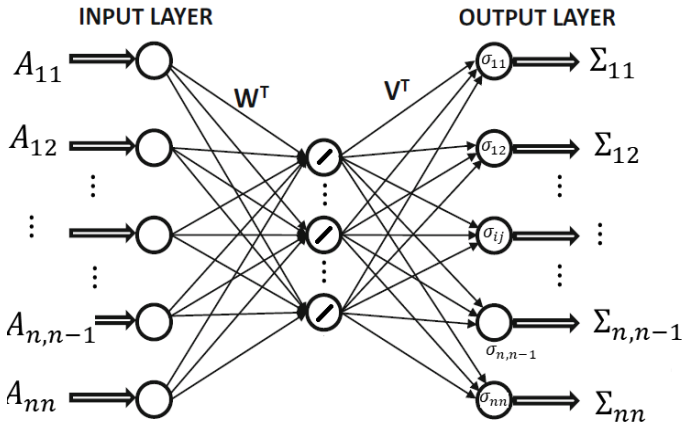
Autoencoder

Consider an autoencoder which has an input layer, a single hidden layer containing the reduced representation, and an output layer with sigmoid units. The hidden layer has linear activation:

- Set up a negative log-likelihood loss function for the case when the input data matrix is known to contain binary values from $0,1$.
- Set up a negative log-likelihood loss function for the case when the input data matrix contains real values from $[0,1]$.

Autoencoder Architecture

Autoencoder which has an input layer, a single hidden layer containing the reduced representation, and an output layer with sigmoid units. The hidden layer has linear activation.



- Hidden layer is linear the autoencoder before sigmoid output translates the input matrix A into UV^T, where $U = W\vec{A}$.
- The output is layer contains sigmoid neurons, so for each element a_{ij} of the input matrix there must be the sigmoid output neuron, that has output $\Sigma_{ij} = \left(\sigma(UV^T)\right)_{ij}$ where Σ is a vector (i.e. matrix treated as a vector) of outputs.

Log likelihood Loss

Set up a negative log-likelihood loss function for the case when the input data matrix is known to contain binary values from $0,1$.

What is Likelihood?

- For Bernoulli variable parameter is proportion/probability of 1's, so likelihood is the conditional pmf for values of these variables $f(x_1, x_2, \dots, x_k | p_1, p_2, \dots, p_k)$.
- Since in our case variables are independent this pmf becomes $f(x_1, x_2, \dots, x_k | p_1, p_2, \dots, p_k) = \prod_{i=1}^{k} f_i(x_i | p_i)$ or in log form $\log f(x_1, x_2, \dots, x_k | p_1, p_2, \dots, p_k) = \sum_{i=1}^{k} \log f_i(x_i | p_i)$.
- So, what is it for variables x_{ij} that take values $a_{ij} \in 0,1$? For each entry a_{ij} pmf is $f(x_{ij} = a_{ij} | p_{ij})$ where p_{ij} is the probability of 1, i.e. $f(a_{ij} | p_{ij}) = \begin{cases} p_{ij} \text{ if } a_{ij} = 1 \\ 1 - p_{ij} \text{ otherwise} \end{cases} = p_{ij}^{a_{ij}} \times (1 - p_{ij})^{1 - a_{ij}}$.
- Thus, when we apply $\log$ the log-likelihood is

$$\log f(a_{ij} | p_{ij}) = \log \left(p_{ij}^{a_{ij}} \times (1 - p_{ij})^{1 - a_{ij}} \right)$$
$$= a_{ij} \log p_{ij} + (1 - a_{ij}) \log (1 - p_{ij})$$

Loss wrt probability p_{ij}

- What is the probability p_{ij} in this case?
  - In fact our modeling is such $a_{ij} \approx (UV^T)_{ij}$ where $(UV^T)_{ij}$ is the pre-activation input of a sigmoid
  - Remember we interpret the output of sigmoid as probability so, $p_{ij} = \left(\sigma(UV^T)\right)_{ij}$ given the pre-activation input of a sigmoid is $(UV^T)_{ij}$

Substituting $p_{ij} = \left(\sigma(UV^T)\right)_{ij}$ into

$$\log f(a_{ij}|p_{ij}) = a_{ij} \log p_{ij} + (1 - a_{ij}) \log(1 - p_{ij})$$

$$= a_{ij} \log\left[\left(\sigma(UV^T)\right)_{ij}\right] + (1 - a_{ij}) \log\left[1 - \left(\sigma(UV^T)\right)_{ij}\right]$$

$$= \begin{cases} \log\left[\left(\sigma(UV^T)\right)_{ij}\right] & \text{if } a_{ij} = 1 \\ \log\left[1 - \left(\sigma(UV^T)\right)_{ij}\right] & \text{if } a_{ij} = 0 \end{cases}$$

Since we are using negative loss we get individual loss for an entry

$$L_{ij} = \begin{cases} -\log\left[\left(\sigma(UV^T)\right)_{ij}\right] & \text{if } a_{ij} = 1 \\ -\log\left[1 - \left(\sigma(UV^T)\right)_{ij}\right] & \text{if } a_{ij} = 0 \end{cases}$$

Clearly total loss is $L = \sum_i^n \sum_j^n L_{ij}$

Fractional Values

Set up a negative log-likelihood loss function for the case when the input data matrix contains real values from $[0,1]$.

As before log-likelihood is

$$\log f(a_{ij}|p_{ij}) = a_{ij} \log p_{ij} + (1 - a_{ij}) \log(1 - p_{ij})$$

where $p_{ij} = \left(\sigma(UV^T)\right)_{ij}$. Substituting into likelihood formula we get

$$\log f\left(a_{ij}\middle|\left(\sigma(UV^T)\right)_{ij}\right) =$$

$$= a_{ij} \log\left[\left(\sigma(UV^T)\right)_{ij}\right] + (1 - a_{ij}) \log\left[1 - \left(\sigma(UV^T)\right)_{ij}\right]$$

But now the value of entry a_{ij} is some fraction F_{ij} so subbing it in into negative individual loss we get

$$L_{ij} = \log f\left(F_{ij}\middle|\left(\sigma(UV^T)\right)_{ij}\right)$$

$$= F_{ij} \log\left[\left(\sigma(UV^T)\right)_{ij}\right] + (1 - F_{ij}) \log\left[1 - \left(\sigma(UV^T)\right)_{ij}\right]$$

As before total loss is $L = \sum_i^n \sum_j^n L_{ij}$

Lecture Overview

1. MT composition
2. Computational Graph
3. Apply convolutional filter
4. Autoencoder
5. **Network Training**
6. True/False and MC questions
7. What happens if …

Same Low Accuracy in Training and Testing

Suppose that you have a model that consistently provides around 80% accuracy on the cross-validation training as well as on the out-of-sample test data. Would you recommend increasing the amount of data or adjusting the model to improve accuracy?

1. Same accuracy on training and testing data suggests that variance is small – because it is the same on different sets of data.
2. Since accuracy is low this suggests that bias is high

So, model is underfitted rather than overfitted. Since we have only 80% accuracy on training data we should try to decrease bias of the model by increasing model complexity – need to increase number of trainable parameters to fit better to the current training data.

Implication: add more hidden non-linear neurons/layers of neurons.

Once we do that we may be moving towards overfitting so we should watch how validation accuracy changes with the change of complexity. Once validation accuracy stop increasing with the increasing complexity we should roll back last changes.

Lecture Overview

1. MT composition
2. Computational Graph
3. Apply convolutional filter
4. Autoencoder
5. Network Training
6. True/False and MC questions
7. What happens if …

T/F

- Skip-gram word2vec takes as input the context and predicts a probability of the word in this context

T/F

Skip-gram word2vec takes as input the context and predicts a probability of the word in this context

- False. Continuous bag of words model does that, skip-gram predicts context from the word

If the data contains a lot of noise then the best way to deal with that is to increase number of layers in order to fit model well

- False. If there is a lot of noise then out of sample data will vary substantially from training data so high variance model will make a lot of errors. We are better off underfitting model rather than overfitting it, so small shallow network is better than deep learner.

MC

In dropout layer (select all that applies)

1. Connection between two nodes can be dropped even when both nodes are present in the network
2. Hidden nodes are dropped with a given probability
3. Connections are dropped only if both nodes are dropped
4. Resulting networks are trained on separate subsampled data sets
5. Weights of resulting networks are averaged and shared before next round

MC

In dropout method (select all that applies)

1. Connection between two nodes can be dropped even when both nodes are present in the network
2. Hidden nodes are dropped with a given probability
3. Connections are dropped only if both nodes are dropped
4. Resulting networks are always trained on separate subsampled data sets
5. Weights of resulting networks are averaged and shared before next round

MC

The following method does not help against vanishing/exploding gradient:

1. Skip connections
2. Freezing hidden-to-hidden weights
3. Limiting backpropagation to segments of the sequences of modest length
4. Using only ReLU neurons as hidden units
5. Using leaky neurons as hidden units

MC

The following method does not help against vanishing/exploding gradient:

1. Skip connections
2. Freezing hidden-to-hidden weights
3. Limiting backpropagation to segments of the sequences of modest length
4. Using only ReLU neurons as hidden units
5. Using leaky neurons as hidden units

Lecture Overview

1. MT composition
2. Computational Graph
3. Apply convolutional filter
4. Autoencoder
5. Network Training
6. True/False and MC questions
7. What happens if …

Change in Accuracy with Size

Does the classification accuracy on the training data generally improve with increasing training data size?

For general classification accuracy we should consider cases of possible differences in training and testing data.

a.) If there is slightly better performance on training data than on testing data and increase of the sample size increases testing accuracy, then training accuracy won't increase. Why?

1. Our parameters were not overused – otherwise performance on training data would be much better than on testing data
2. Parameters could be underused. If so, what happens with training accuracy with increasing data size?
   - Nothing: since the parameters were underused they were fitted perfectly for old training data just as they will be for increased in size new training data
3. Parameters could have been used perfectly. Increase in data size will overfit NN

Change in Accuracy with Size

Does the classification accuracy on the training data generally improve with increasing training data size?

For general classification accuracy we should consider cases of possible differences in training and testing data.

b.) If there is much better performance on training data than on testing data and the increase of the size of the training data does not result in increase of testing accuracy then it means that we started with already overfitted model.

- so if we add the size to training data, variance over the subsets of data for this model will only increase – model will try to adapt by memorizing more data, but because it was already overfitted it meant that parameters were already overused, so increase in training data size will decrease accuracy on each smaller subset the training data, i.e. increase variance without improving bias. Thus on overall data accuracy will worsens, because the model cannot memorize any more data with a given number of parameters.

Change in Accuracy with Size

Does the classification accuracy on the training data generally improve with increasing training data size?

For general classification accuracy we should consider cases of possible differences in training and testing data.

c.) If performance on the training data is the same or slightly worse than on the testing data and is not very good then we have the case of high bias-low variance.

- Increasing training data size will not increase the training accuracy since the model is incapable to/does not have enough parameters for fitting the data. Which means that on a bigger data set it still won't be able to do so. Consequently training accuracy will not increase.

Change in Accuracy with Size

Does the classification accuracy on the training data generally improve with increasing training data size?

Conclusion: Training accuracy does not generally increase with increasing data size.

This fact may lead you to think that testing accuracy will not increase either. This is incorrect test accuracy should increase with increasing training data size and if it does not then the model is bad or we are already at maximum fitting capacity

Change in Average Point-wise Accuracy w/Size

- How about the point-wise average of the loss on training instances? Explain your answer.
- The average loss must increase with training data size if a model has low bias, high variance, as the model becomes less specific to particular training instances.
- If a model has high bias and low variance then per instance loss will not change

When Testing accuracy = Training Accuracy?

At what point will training and testing accuracy become similar?

Again, consider bias and variance.

- If the model achieved bias/variance tradeoff (well fitted) and the size of training data is very large, the training and testing accuracy normally are similar because training data represents well general population.
- If the model is underfitted the situation is the same as before – accuracy on training and testing data when training data is large is the same.
- What means that the model is overfitted when training data is large? So large data is still not representative of variation in the general population. This happens even with large data in which case training accuracy will be better than testing.
  - But normally large data means that this data is representative which implies that overfitting to it means fitting well to general population. In which case testing and training accuracy become similar.